Manuscript on "Empowering Latent Variable Model with Nonlinearity By Linear Differential Equations"

Zhichao Chen[1]
[1]Zhejiang University

Keywords: Latent Variable Model, Machine Learning, Inferential Sensors, Bayeisan Inference

1. Introduction

However, leveraging the specific properties of linear models and deriving corresponding algorithms for learning LLVMs with DE pose significant challenges. A traditional method for learning the LLVM is expectation maximization (EM) algorithm [1, 2]. For the given observations sampled from the LVMs with unknown parameters, the EM algorithm aims to find the maximum likelihood estimation of the model parameters in an alternative manner. In its first step (E-step), the EM algorithm estimates a function of the expected values of the latent variables. Subsequently in the second step (M-step), it finds the maximum likelihood parameter estimates. Based on abovementioned content, the parameter learning for such diffusion process-based latent variable model remains following challenges:

1. **Large scale posetrior computation (E-Step):** In E-step, the posterior process inference needs to be performed for each sample. However, due to the discretization of time domain in diffusion process, the inference of the continuous-time posterior process incurs substantial storage and computational costs.
2. **Gradient with respect to model parameters (M-Step):** In the M-step, obtaining expressions for the learning objective gradient with respect to the model parameters becomes challenging due to the presence of differential equations.
3. **Model convergence:** The convergence of the parameter learning process is a crucial consideration, particularly whether the alternative iteration process of the EM algorithm still converges for such diffusion process-based LLVM.

The contributions of this paper are summarized as follows:

1. This paper proposed a novel NPLVM by introducing the linear diffusion process as data generative process, and derive its corresponding learning objective *ab initio*.
2. Based on the Pontryagin maximum principle, two constrainted programming problem was proposed, which avoid the discretization of the diffusion process. On this basis, the expressions of the gradient with respect to the model parameter are also obtained.
3. Furthermore, the convergence of the proposed method is analyzed in theory and practice.

The remainder of this paper is structured as follows: To better understand this paper, concerning background knowledge is introduced in Section 2. In section 3, the preliminaries are introduced to better understand the proposed method. In section 4, the learning algorithm is derived in detailed from the perspecive of optimization & optimal control principles. Besides, the parameter learning

Corresponding author(s): songzhihuan@zju.edu.cn

algorithm and convergence are also discussed in this section. To demonstrate the effectiveness of the proposed method, various experiments are conducted in section 5. Furthermore, the conclusions and future research directions are summarized in section 6.

2. Preliminaries

2.1. Adjoint State in OC problem

Suppose we have an OC problem given as follow:

$$\begin{aligned} &\min \ \ \phi(x_{t_f}) \\ &s.t. \quad \begin{cases} \mathcal{F}(\frac{\mathrm{d}x_\tau}{\mathrm{d}\tau}, x_\tau, \theta, \tau) = \frac{\mathrm{d}x_\tau}{\mathrm{d}\tau} - f(x_t, \theta, \tau) \\ x_{t_0} = x_{\text{init}} \\ t_0 < t_f \end{cases}, \end{aligned} \tag{1}$$

Based on the celebrated Lagrangian multiplier method, the problem can be reformulated as follow:

$$\psi = \phi(x_{t_f}) + \int_{t_0}^{t_f} \lambda_\tau^\top \mathcal{F}(\frac{\mathrm{d}x_\tau}{\mathrm{d}\tau}, x_\tau, \theta, \tau)\mathrm{d}\tau, \tag{2}$$

where λ is known as Lagrangian multiplier/ adjoint. Notably, λ satisfies the following equation:

$$\frac{\mathrm{d}\lambda_\tau}{\mathrm{d}\tau} = -\lambda_\tau \frac{\partial f}{\partial x}, \quad \text{B.C. } \lambda_{t_f} = \frac{\partial \phi(x_{t_f})}{\partial x_{t_f}}, \tag{3}$$

where B.C. is the abbreviation of boundary condiiton.

2.2. Alternative Optimization Framework

Consider a special case where the decision variables w and v in the objective function term are separable (assume f and h are convex):

$$\begin{aligned} \min \quad & \text{Obj}(v, w) = f(w) + h(v) \\ s.t. \quad & Aw + Bv = c \end{aligned}. \tag{4}$$

According to the augmented Lagrangian multiplier method [3], we can cast (4) to an unconstrained optimization problem, with the objective function:

$$\begin{aligned} L = & f(w) + h(v) + \lambda^\top (Aw + Bv - c) \\ & + \frac{\rho}{2} \|Aw + Bv - c\|_2^2, \end{aligned} \tag{5}$$

where ρ is quadratic penalty coefficient. The celebrated Alternating Direction Method of Multipliers (ADMM) algorithm [3] is a common algorithm to solve (5), where variables v and w are optimized separately at each iteration:

$$\begin{cases} w^{k+1} & = \arg\min_w L(w, v^k, \lambda) \\ v^{k+1} & = \arg\min_v L(w^{k+1}, v, \lambda) \\ \lambda^{k+1} & = \lambda^k + \rho(Aw^{k+1} + Bv^{k+1} - c) \end{cases}. \tag{6}$$

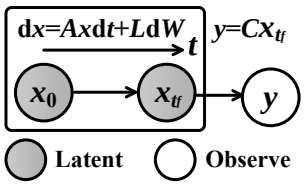

Figure 1 | Model Structure

3. Model Derivation

At first, consider a probabilistic space denoted by a 3-element tuple $(\Omega, \mathcal{F}, \mathbb{P})$ [4], where Ω is sample space, σ-algebra $\mathcal{F}$ is set of events, and $\mathbb{P}$ is probability measure: $\mathcal{F} \mapsto [0, 1]$. Define an $\mathcal{F}_\tau$-adapted adapted Wiener process $W(\tau)$ on this probabilistic space.

On this basis, under probabilistic measure $\mathbb{P}$, we consider the model structure as Fig. 1 shown. We assume that our dimension-H observation data y is generated via (7) from dimension-D state x_{t_f}:

$$y = Cx_{t_f}, \tag{7}$$

where final state x_{t_f} is generated via a linear diffusion process defined as (8) after a predefined time t_f (t_f is always set as 1 in the context of generative models [5, 6]):

$$x_{t_f} = x_0 + \int_0^{t_f} Ax\mathrm{d}\tau + \int_0^t L\mathrm{d}W(\tau), \tag{8}$$

and A, L, and $W(\tau)$ is referred to as drift term, volatility term, and Winner process, respectively. To learn this model, we introduce an approximation process defined in (9) under $\mathbb{Q}$ measure:

$$x_{t_f} = x_0 + \int_0^{t_f} \tilde{A}x\mathrm{d}\tau + \int_0^t L\mathrm{d}W(\tau). \tag{9}$$

On this basis, we have following lower bound of the data log-likelihood estimation:

Lemma 1. *The lower bound named Evidence Lower BOund (ELBO) $\mathcal{L}$ of data log-likelihood algorithm is obtained as follow:*

$$\begin{aligned} \log p(y) &\geq -\mathbb{D}_{\mathrm{KL}}(\mathbb{Q}\|\mathbb{P}) + \mathbb{E}_{\mathbb{Q}}[\log p(y|x_{t_f})] \triangleq \mathcal{L} \\ &\Longleftrightarrow \arg\max \log p(y) = \arg\max \mathcal{L}, \end{aligned} \tag{10}$$

where $\mathbb{D}_{\mathrm{KL}}(\mathbb{Q}\|\mathbb{P})$ is Kullback-Leiber divergence (KL divergence), which is defined as follow:

$$\mathbb{D}_{\mathrm{KL}}(\mathbb{Q}\|\mathbb{P}) = \int \log \frac{\mathrm{d}\mathbb{Q}(x)}{\mathrm{d}\mathbb{P}(x)} \mathrm{d}\mathbb{Q}, \tag{11}$$

and the derivative is named Radon-Nikodym derivative.

Maximizing the model log-likelihood $\log p(y)$ is equivalence to the maximizing the ELBO $\mathcal{L}$. And thus, the rest of this subsection will focus on the simplification of ELBO. We first focus on the KL divergence term. To simplify the KL divergence, we will introduce the celebrated Girsanov theorem and Radon-Nikodym theorem as follow:

To simplify the KL divergence, we introduce the likelihood ratio of two diffusion processes as follow:

Theorem 3.1 (Likelihood ratio of Itô Process [7]). *Assume we have two diffusion processes defined as follow:*

$$\begin{cases} \mathrm{d}x &= f(x,t)\mathrm{d}\tau + \mathrm{d}W, x_0 = x_{\text{init}} \\ \mathrm{d}y &= g(y,t)\mathrm{d}\tau + \mathrm{d}W, y_0 = x_{\text{init}} \end{cases}. \tag{12}$$

The Radon-Nikodym derivative along their respective path measures $\mathbb{Q}$ and $\mathbb{P}$ is given by

$$\begin{aligned} \frac{\mathrm{d}\mathbb{Q}}{\mathrm{d}\mathbb{P}}(x) = \exp(-\frac{1}{2}\int_0^t \|g(x,\tau) - f(x,\tau)\|^2 \mathrm{d}\tau \\ + \int_0^t (g(x,\tau) - f(x,\tau))^\top \mathrm{d}W). \end{aligned} \tag{13}$$

Based on this theorem, plugging (13) into (10), we therefore obtain following learning objective:

$$\arg\min_{A,C,x_0,u} \mathbb{E}_{\mathbb{Q}}[-\log p(y|x_{t_f}) + \frac{1}{2}\int_0^{t_f} \|u\|_2^2 \mathrm{d}\tau], \tag{14}$$

and u is given by follow equation:

$$u = \tilde{A}x - Ax. \tag{15}$$

We lies on the fact that the fact that Itô integral $\int_0^{t_f} u^\top \mathrm{d}W$ is a martingale. In other words, $\mathbb{E}_{\mathbb{Q}} = \int_0^{t_f} u^\top \mathrm{d}W = 0$ holds.

Note that, (14) is a stochastic optimal control problem, we want to simplify this problem into a deterministic optimal control problem and obtain the paraeter as fast as possible. To this end, we have following proposition:

Proposition 3.2. *Assume the final state* $\log p(y|x_{t_f})$ *is conceptualized as a Dirac delta distribution (i.e.,* $\delta(y - Cx_{t_f})$*) and approximated by Gaussian distribution* $\mathcal{N}(Cx_{t_f}, \Sigma_y)$*. If we concentrate on the mean value of learning objective, the proposed stochastic optimal control can be simplified into the following optimal control (OC) problem:*

$$\begin{aligned} \min_{\nu,\theta} \ & \frac{1}{2}(y - C\mu_{t_f})^\top \Sigma_y^{-1}(y - C\mu_{t_f}) + \frac{1}{2}\int_0^{t_f} \|\nu\|_2^2 \mathrm{d}\tau \\ s.t. \ & \begin{cases} \frac{\mathrm{d}\mu}{\mathrm{d}\tau} = A\mu + L\nu \\ \mu_0 = \mu_{\text{init}} \end{cases} \end{aligned}, \tag{16}$$

where D-dimensional vector μ is the mean value of state ($\mathbb{E}(x)$), D-dimensional vector ν is the mean value control policy ($\mathbb{E}(u)$), $\theta = \{A, \mu_{\text{init}}, C\}$ is the set of parameters to be identified, τ stands for time, $A \in \mathbb{R}^{D\times D}$ is the state transition matrix, diagnoal matrix $L \in \mathbb{R}^{D\times D}$ is control coefficient matrix, $C \in \mathbb{R}^{H\times D}$ is observation matrix, H is the observation dimension, $\Sigma_y \in \mathbb{R}^{H\times H}$ is covariance of observation data, init is abbreviation of initial.

Proof. Based on the principle of stochastic differential equation, the following ordinary differential equation about the mean can be obtained:

$$\mathbb{E}[\mathrm{d}x] = \mathbb{E}[Ax\mathrm{d}\tau + L\nu + LdW_t]. \tag{17}$$

As such, the mean value transition function can be obtained:

$$\frac{\mathrm{d}\mu}{\mathrm{d}\tau} = A\mu + L\nu. \tag{18}$$

Besides, the objective function can be obtained as follow with Gaussian assumption:

$$\begin{aligned}&\mathbb{E}_{\mathbb{Q}}[\log p(y|x)]\\=&\mathbb{E}_{\mathbb{Q}}[(y-Cx_{t_f})^\top\Sigma_y^{-1}(y-Cx_{t_f})]\\=&(y-C\mu_{t_f})^\top\Sigma_y^{-1}(y-C\mu_{t_f})+Tr(\Sigma_y^{-1}\Sigma_\mu),\end{aligned} \tag{19}$$

$$\int_0^{t_f}\|u\|_2^2\mathrm{d}\tau=\int_0^{t_f}\nu^2+\Sigma_u\mathrm{d}\tau. \tag{20}$$

Therefore, the objective function can be decomposed into the mean and covariance term, respectively:

$$\begin{aligned}&\mathbb{E}_{\mathbb{Q}}[\log p(y|x)-\frac{1}{2}\int_0^{t_f}\|u\|_2^2\mathrm{d}\tau]\\=&-[(y-C\mu_{t_f})^\top\Sigma_y^{-1}(y-C\mu_{t_f})\\&+Tr(\Sigma_y^{-1}\Sigma_\mu)]-\int_0^{t_f}[\nu^2+\Sigma_u]\mathrm{d}\tau.\end{aligned} \tag{21}$$

Note that, we can mainly concentrate on the mean value, and the learning objective can be given in (22).

$$\mathcal{L}=\frac{1}{2}(y-C\mu_{t_f})^\top\Sigma_y^{-1}(y-C\mu_{t_f})+\frac{1}{2}\int_0^{t_f}\nu^2\mathrm{d}\tau \tag{22}$$

□

On this basis, in the following section, we want to solve this OC problem and derive our parameter learning algorithm.

4. Adjoint-EM for Parameter Estimation

4.1. Inspiration of Parameter Learning Algorithm

In this section, we derive the parameter estimation for the proposed model introduced in Section 3. Leveraging Proposition 3.2, we compare the learning objective presented in (16) with the learning objective in (4). From this comparison, we observe that the objective can be expressed as follows:

$$\underbrace{\frac{1}{2}(y-C\mu_{t_f})^\top\Sigma_y^{-1}(y-C\mu_{t_f})}_{f(w)}+\underbrace{\frac{1}{2}\int_0^{t_f}\|\nu\|_2^2\mathrm{d}\tau}_{h(v)}, \tag{23}$$

where ν corresponds to v, and θ corresponds to w. Consequently, we can employ the celebrated ADMM algorithm to solve for parameter estimation, optimizing ν and θ alternately.

Furthermore, it is noteworthy that optimizing the optimal control policy ν bears resemblance to solving the posterior process, akin to the E-step in the EM algorithm introduced in Section **??**. On the other hand, the parameter optimization is analogous to the M-step in the EM algorithm introduced in Section **??**. Consequently, we can consider our algorithm as a specialized variant of the EM algorithm. Additionally, it's important to highlight that our model consistently satisfies the constraint stated in (16), ensuring that $\frac{\mathrm{d}\mu}{\mathrm{d}\tau}-A\mu-L\nu=0$ always holds true. With this in mind, the subsequent subsections will not delve into the coefficient ρ.

4.2. E-Step: Large-Scale OC policy Computation with Adjoint State

Intuitively, the inner optimal control problem defined by (4) can be reformulated as an optimization problem of Differential Algebraic Equations (DAEs) [8]. Consequently, we can employ a conventional nonlinear programming method by discretizing μ, ν, and $|\nu|_2^2$ over the time domain $[0, t_f]$ in order to solve it and obtain the policy ν. However, this approach leads to the introduction of numerous intermediate variables and equation constraints at the collocation points, thereby resulting in an inevitable increase in the scale of our optimization problem. This issue becomes more pronounced when attempting to apply this optimal control problem to a large number of training samples. To illustrate this phenomenon, we present the following propositions concerning the time complexity:

Proposition 4.1. *Suppose we employ the finite difference method (FDM) to discretize the optimal control problem defined by* (16)*, and considering a total of* $p+1$ *points in the time domain* $[0, t_f]$ *(including* 0 *and* t_f*) to discreteize our model. If we set the iteration accuracy for the optimal control problem defined by* (16) *as* ϵ*, the lower bound of our time complexity is* $O\left([(p+1)(3D+2)+H]^{3.5}\frac{1}{\epsilon}\right)$*. When we apply this strategy to a training sample of size* n*, the lower bound of our time complexity becomes* $O\left(\{n[(p+1)(3D+2)+H]\}^{3.5}\frac{1}{\epsilon}\right)$.

The aforementioned proposition clearly demonstrates that discretization of the DAEs is not feasible for parameter learning, as it leads to scalability issues with increasing data numbers. Therefore, finding a way to avoid discretization in the time domain is crucial for computing the OC policy. In order to achieve this, we introduce the following lemma to obtain the OC policy:

Lemma 2. *For the optimal control defined in* (16)*, the OC policy* ν^* *is given as follow:*

$$\nu^* = -L\lambda, \tag{24}$$

where D*-dimensional vector* λ *is defined as the adjoint (also known as co-state), and satisfies following differential equation and boundary condition* (B.C.) *at time* t_f*:*

$$\begin{aligned} &\frac{\mathrm{d}\lambda}{\mathrm{d}\tau} = -A^\top \lambda \\ \text{B.C.}\quad &\lambda_{t_f} = C^\top \Sigma_y^{-1}(C\mu_{t_f} - y) \end{aligned}. \tag{25}$$

Unfortunately, the boundary condition (B.C.) at t_f remains unknown because the state μ is influenced by λ due to the presence of the control policy in the state transition equation. Consequently, even if we have knowledge of the differential equation, we are still unable to determine the optimal policy. In order to address this challenge, our aim is to develop a discretization-free algorithm to solve the adjoint at time t_0. To achieve this, we present the following proposition for solving the adjoint at time t_0:

Proposition 4.2. *The adjoint* λ *at time* 0 *can be solved by the following quadratic programming problem:*

$$\begin{aligned} &\arg\min_{\lambda_0} \ \|\Gamma - \Omega\xi_0\|_2 \\ &s.t. \quad \begin{cases} \Gamma = C^\top \Sigma_y^{-1} y \\ \Omega = \begin{bmatrix} C^\top \Sigma_y^{-1} C & -I \end{bmatrix} \exp\left(\begin{bmatrix} A & -LL \\ \mathbf{0}_{D\times D} & -A^\top \end{bmatrix} t_f\right) \\ \xi_0 = [\mu_0^\top, \lambda_0^\top]^\top \\ \mu_0 = \mu_{\text{init}}, \end{cases} \end{aligned}, \tag{26}$$

where matrix $\Omega \in \mathbb{R}^{D\times 2D}$*,* $\Gamma \in \mathbb{R}^{D\times 1}$*, and* $\xi_0 \in \mathbb{R}^{2D\times 1}$.

While proposition 4.2 successfully converts the optimal control problem into a discretization-free constrained quadratic programming problem, solving this problem still requires the use of a nonlinear programming solver. In order to eliminate the reliance on such a solver, we can further transform this constrained quadratic programming problem into a least squares problem, as outlined in the following proposition:

Proposition 4.3. *For the constrainted programming problem defined in (26), we can obtain the global optimal value of λ_0 by solving the following least square problem:*

$$\arg\min_{\lambda_0} \left\| \underbrace{\begin{bmatrix} \Omega^\top\Omega & I_{D\times D} \\ I_{D\times D} \quad \mathbf{0}_{D\times D} & \mathbf{0}_{D\times D} \end{bmatrix}}_{\Xi} \underbrace{\begin{bmatrix} \mu_0 \\ \lambda_0 \\ \alpha \end{bmatrix}}_{\Lambda} - \underbrace{\begin{bmatrix} \Omega^\top\Gamma \\ \mu_0 \end{bmatrix}}_{\Delta} \right\|_2, \tag{27}$$

where $\alpha \in \mathbb{R}^{D\times 1}$ is Lagrangian multiplier for quadratic programming problem defined in (26). Based on this conversion, the initial adjoint λ_0 can be solved by following equation:

$$\lambda_0 = \Lambda^*[D:\ 2D], \tag{28}$$

where $[D:\ 2D]$ indicates that select the elements from position D to position $2D$, and Λ^* can be solved by sigular value decomposition (SVD) algorithm or pesudo-inverse of matrix Ξ.

Remark 4.1. *The least square problem defined in (27) has a unique solution since matrix Ξ is full-ranked.*

Remark 4.2. *By solving the least square problem defined in (27), our OC policy computational time complexity reduce to $O(3D^3 + D^2 + D)$ under the Truncated SVD algorithm [9]. When we scale it up to a total amount of n training samples, the time complexity is still $O(3D^3 + D^2 + D)$ since matrix Ξ is invariant of observation data.*

Through the aforementioned conversions, we have successfully devised an efficient approach for solving the E-step computation problem, which exhibits linear time complexity. In the subsequent subsection, we will outline our proposal for parameter gradient estimation.

4.3. M-Step: Adjoint-Based Parameter Gradient

Based on the optimal ν^* obtained in previous subsection, this subsection will focus on the optimization of parameters θ. In this subsection, we propose the gradient value with respect to parameter A, C, and μ_{init}, respectively.

Similar to Neural ODE model [10], we can obtain our gradient with respect to parameters with the help of adjoint state λ. For simplicity, we use estimate the gradient based on maximum likelihood estimation (MLE) i.e. regardless of the $\frac{1}{2}\int_0^{t_f} \|\nu\|_2^2 \mathrm{d}\tau$. We can obtain the concerning Hamiltonian function as follow:

$$\begin{aligned} \mathcal{H}_{\text{MLE}} = &\frac{1}{2}(y - C\mu_{t_f})^\top \Sigma_y^{-1}(y - C\mu_{t_f}) \\ &- \int_0^{t_f} \lambda^\top \left(\frac{\mathrm{d}\mu}{\mathrm{d}\tau} - A\mu + L\nu\right)\mathrm{d}\tau. \end{aligned} \tag{29}$$

Note that, the integral term can be reformulated as follow:

$$\begin{aligned}&\int_0^{t_f} \lambda^\top (\frac{\mathrm{d}\mu}{\mathrm{d}\tau} - A\mu - L\nu)\mathrm{d}\tau \\ =&\lambda^\top \mu|_0^{t_f} - \int_0^{t_f} \mu^\top \frac{\mathrm{d}\lambda}{\mathrm{d}\tau} - \int_0^{t_f} \lambda^\top (A\mu + L\nu)\mathrm{d}\tau \\ =&\lambda_{t_f}^\top \mu_{t_f} - \lambda_0^\top \mu_0 - \int_0^{t_f} [\mu^\top \frac{\mathrm{d}\lambda}{\mathrm{d}\tau} + \lambda^\top (A\mu + L\nu)]\mathrm{d}\tau.\end{aligned} \tag{30}$$

On this basis, the Gradient with respect to parameter θ can is derived in the following subsections.

4.3.1. Gradient with respect to C

We can directly obtain the gradient as follow:

$$\begin{aligned}&\frac{\partial \mathcal{L}}{\partial C} \approx \frac{\partial \mathcal{H}_{\text{MLE}}}{\partial C} \\ =&\frac{1}{2}\frac{\partial (y - C\mu_{t_f})^\top \Sigma_y^{-1}(y - C\mu_{t_f})}{\partial C} \\ =&\Sigma_y^{-1}(C\mu_{t_f} - y)\mu_{t_f}^\top.\end{aligned} \tag{31}$$

4.3.2. Gradient with respect to μ_0

Based on (30), the gradient of the state transition equation with respect to μ_0 can be derived as follow:

$$\begin{aligned}&\frac{\mathrm{d}}{\mathrm{d}\mu_0}[\int_0^{t_f} \lambda^\top (\frac{\mathrm{d}\mu}{\mathrm{d}\tau} - A\mu - L\nu)\mathrm{d}\tau] \\ =&\frac{\mathrm{d}\mu_{t_f}}{\mathrm{d}\mu_0}\lambda_{t_f} - \lambda_0 - \int_0^{t_f} \frac{\partial \mu}{\partial \mu_0}(\frac{\mathrm{d}\lambda}{\mathrm{d}\tau} + A^\top \lambda)\mathrm{d}\tau.\end{aligned} \tag{32}$$

Plug (29) into (32), we can obtain the following equation:

$$\begin{aligned}&\frac{\partial \mathcal{L}}{\partial \mu_0} \\ \approx&\frac{\partial \mathcal{H}_{\text{MLE}}}{\partial \mu_0} \\ =&\frac{\partial \mathcal{L}}{\partial \mu_0} - \frac{\partial}{\partial \mu_0}[\int_0^{t_f} \lambda^\top (\frac{\mathrm{d}\mu}{\mathrm{d}\tau} - A\mu + L\nu)\mathrm{d}\tau] \\ =&\frac{\mathrm{d}\mu_{t_f}}{\mathrm{d}\mu_0}(\frac{\partial \mathcal{L}}{\partial \mu_{t_f}} - \lambda_{t_f}) + \int_0^{t_f} (\frac{\mathrm{d}\lambda}{\mathrm{d}\tau} + A^\top \lambda)\mathrm{d}\tau + \lambda_0 \\ =&\lambda_0 \\ =&\lambda_{t_f} - \int_{t_f}^0 A^\top \lambda \mathrm{d}\tau,\end{aligned} \tag{33}$$

where last line lies on the expression of the adjoint equation and its corresponding boundary condition defined in (25).

4.3.3. *Gradient with respect to* A

Based on (30), the gradient with respect to parameter A can be obtained as follow:

$$\begin{aligned}
&\frac{\mathrm{d}}{\mathrm{d}A}[\int_0^{t_f}\lambda^\top(\frac{\mathrm{d}\mu}{\mathrm{d}\tau}-A\mu+L\nu)\mathrm{d}\tau]\\
=&\lambda_{t_f}^\top\frac{\mathrm{d}\mu_{t_f}}{\mathrm{d}A}-\int_0^{t_f}(\frac{\mathrm{d}\lambda}{\mathrm{d}\tau}\frac{\mathrm{d}\mu}{\mathrm{d}A}^\top+\lambda\frac{\mathrm{d}(A\mu+L\nu)}{\mathrm{d}A}^\top)\mathrm{d}\tau\\
=&\lambda_{t_f}^\top\frac{\mathrm{d}\mu_{t_f}}{\mathrm{d}A}-\int_0^{t_f}\lambda\mu^\top\mathrm{d}\tau\\
&-\int_0^{t_f}[\frac{\mathrm{d}\lambda}{\mathrm{d}\tau}+\frac{\partial(A\mu+L\nu)}{\partial\mu}\lambda]\frac{\mathrm{d}\mu}{\mathrm{d}A}^\top\mathrm{d}\tau.
\end{aligned}\tag{34}$$

Plug (34) into (29), we can get:

$$\begin{aligned}
&\frac{\partial\mathcal{L}}{\partial A}\\
\approx&\frac{\partial\mathcal{H}_{\mathrm{MLE}}}{\partial A}\\
=&(\frac{\partial\mathcal{L}}{\partial\mu_{t_f}})^\top\frac{\mathrm{d}\mu_{t_f}}{\mathrm{d}A}-\frac{\mathrm{d}}{\mathrm{d}A}[\int_0^{t_f}\lambda^\top(\frac{\mathrm{d}\mu}{\mathrm{d}\tau}-A\mu-L\nu)\mathrm{d}\tau]\\
=&[(\frac{\partial\mathcal{L}}{\partial\mu_{t_f}})^\top-\lambda_{t_f}^\top]\frac{\mathrm{d}\mu_{t_f}}{\mathrm{d}A}\\
&+\int_0^{t_f}(\frac{\mathrm{d}\lambda}{\mathrm{d}\tau}+\frac{\partial(A\mu+L\nu)}{\partial\mu}\lambda)\frac{\mathrm{d}\mu}{\mathrm{d}\tau}^\top\mathrm{d}\tau+\int_0^{t_f}\lambda\mu^\top\mathrm{d}\tau,\\
=&\underbrace{[C^\top\Sigma_y^{-1}(C\mu_{t_f}-y)-\lambda_{t_f}]^\top}_{0}\frac{\mathrm{d}\mu_{t_f}}{\mathrm{d}A}\\
&+\int_0^{t_f}(\underbrace{\frac{\mathrm{d}\lambda}{\mathrm{d}\tau}+A^\top\lambda}_{0})\frac{\mathrm{d}\mu}{\mathrm{d}\tau}^\top\mathrm{d}\tau+\int_0^{t_f}\lambda\mu^\top\mathrm{d}\tau\\
=&\int_0^{t_f}\lambda\mu^\top\mathrm{d}\tau.
\end{aligned}\tag{35}$$

where the first zero in the fourth line corresponds to the boundary condition of the adjoint state as defined in (25), and the second zero in the fourth line corresponds to the transition equation of the adjoint state as given in (25).

4.4. Overall Summary & Concerning Analysis

Based on the abovementioned derivation, we can see that the E step and M step are highly rely on the adjoint λ, and thus we named our algorithm "Adjoint EM". On this basis, the algorithm is summarized as Algorithm 1 shown.

Proposition 4.4. *The convergence*[1] *of the optimization process in Algorithm 1 concerning the loss function $\mathcal{L}$ can be ensured under the following condition: The learning rate η in the M-Step is sufficiently small.*

[1] In accordance with the convergence definition used in the conventional EM algorithm [11], convergence is achieved when the loss function exhibits monotonic decrease and remains bounded during the iteration process.

Algorithm 1 LDE-NLVM Learning with Adjoint EM

Input: Training Data: $\{y^{(i)}\} \in \mathcal{D}_{train}$
Hyperparameters: Latent Variable Dimension: d_{latent}, Learning Rate: η, Iteration Epoch: E, Initial guess of parameters: $\theta = \{A, C, \mu_{\text{init}}\}$, Solving accuracy ϵ, and end time t_f.

Output: Trained parameter $\theta = \{A, C, \mu_{init}\}$.

Set $k = 1$, $\mathcal{L}_0 = \infty$. Calculate the data covariance matrix Σ_y ;
while $k < E$ **do**
  Solve the LS problem defined in (27), and obtain λ_0;
  Solve the state μ_{t_f}, and estimate the $\mathcal{L}$ as follow:
  $\mathcal{L}_k \leftarrow \frac{1}{2}(y - C\mu_{t_f})^\top \Sigma_y^{-1}(y - C\mu_{t_f}) + \frac{1}{2}\int_0^{t_f} \nu^2 \mathrm{d}\tau$;
  Update the parameter as follows:
  $C \leftarrow C - \eta \times C^\top \Sigma_y^{-1}(C\mu_{t_f} - y)$
  $A \leftarrow A - \eta \times \int_0^{t_f} \lambda x^\top \mathrm{d}\tau$
  $\mu_{\text{init}} \leftarrow \mu_{\text{init}} - \eta \times \lambda_0$
  while $|\mathcal{L}_k - \mathcal{L}_{k-1}| < \epsilon$ **do**
    Break the loop
  end while
end while
return θ

Proof. Suppose the ELBO $\mathcal{L}$ at iteration time k to be $\mathcal{L}_k$, and the loss during the E-step at time k to be $\mathcal{L}_{k+\frac{1}{2}}$. The following inequality holds:

$$\mathcal{L}_{k+\frac{1}{2}} \leq \mathcal{L}_k. \tag{36}$$

It is not hard to understand why this inequality holds: In the E-step we are obtaining the optimal control policy, and proposition 2 proves that the optimal control signal will give the minimum value of $\mathcal{L}$.

Meanwhile, in the M-step, the parameter is updated based on the gradient method. And thus, we can give the iterative equation for parameter set θ:

$$\theta_{\kappa+1} = \theta_{\kappa+\frac{1}{2}} - \eta \nabla_\theta \mathcal{L}, \tag{37}$$

Denote the η as $\Delta\kappa$ and consider $\Delta\kappa \to 0$, we can reformulate (37) as follow:

$$\frac{\mathrm{d}\theta}{\mathrm{d}\kappa} = -\nabla_\theta \mathcal{L}. \tag{38}$$

Compare (38) with (37), we can see that (37) is obtained by discretizing (38) via Euler's method [12]. On this basis, we can obtain the following differential equation for $\mathcal{L}$:

$$\frac{\mathrm{d}\mathcal{L}}{\mathrm{d}\kappa} = \left\langle \nabla_\theta \mathcal{L}, \frac{\mathrm{d}\theta}{\mathrm{d}\kappa} \right\rangle = -\|\nabla_\theta \mathcal{L}\|^2 \leq 0. \tag{39}$$

From (39), it becomes evident that the method based on gradient descent is capable of progressively reducing $\mathcal{L}$. Building upon this foundation, a lower learning rate, which corresponds to the step size in

Euler's method, results in the iteration curve of $\mathcal{L}$ more closely approximating the ODE defined in (39). Consequently, a lower learning rate η yields a sequence of the learning objective that monotonically decreases. On this basis, if the learning rate is small enough, the following inequality will holds:

$$\mathcal{L}_{k+1} \leq \mathcal{L}_{k+\frac{1}{2}}. \tag{40}$$

Plug (40) into (36), we can obtain the following result:

$$\mathcal{L}_{k+1} \leq \mathcal{L}_k, \tag{41}$$

which indicates that the loss function is monotonicity decreasing. Besides, the loss function will be bounded by the log-likelihood function due to the existence of the non-negative control energy term:

$$\mathcal{L} \geq (y - C\mu_{t_f})^\top \Sigma_y^{-1} (y - C\mu_{t_f}). \tag{42}$$

Based on (41) and (42), the convergence of the algorithm is guaranteed according to the celebrated monotone convergence theorem. □

5. Experiments

5.1. Problem Solving Protocols

In our experiments, we utilized IpOpt [13], a well-known large-scale interior point method-based nonlinear programming solver, to solve our DAEs and QP problems. All optimization problems were implemented using the pyomo platform [14], and we employed the FDM with 2.0×10^{-3} as step size for DAEs [8] discretization. The solving accuracy for ϵ in all optimization problems was set to 1.0×10^{-9}. Our models were evaluated on a laptop with 16 GB of RAM and an AMD Xeon CPU.

5.2. Effectiveness of OC policy Computation

In this subsection, the effectiveness of the proposed equivalence conversion for the optimal control problem is evaluated by addressing the question: 'Is the conversion of the DAE problem effective?'. To illustrate this, the following dynamical system is selected:

$$\begin{aligned} \min_{\nu} \ & \frac{1}{2}(y - C\mu_1)^\top \Sigma_y^{-1} (y - C\mu_1) + \frac{1}{2}\int_0^1 \|\nu\|_2^2 \mathrm{d}\tau \\ s.t. \ & \begin{cases} \frac{d\mu}{dt} = \begin{bmatrix} 1 & 2 & 2 \\ 0 & 3 & 5 \\ 0 & 0 & 1 \end{bmatrix} \mu + \begin{bmatrix} 3 & 2 & 1 \\ 0 & 5 & 4 \\ 0 & 0 & 1 \end{bmatrix} \nu \\ \hat{y} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \end{bmatrix} \mu \\ \mu_{\text{init}} = \begin{bmatrix} 0 & 0 & 0 \end{bmatrix}^\top \\ y_1 = \begin{bmatrix} 10 & 10 \end{bmatrix}^\top + e, e \sim \mathcal{N}(0, I_{2\times 2}) \end{cases} \end{aligned} \tag{43}$$

Fig. 2 presents the results of the computation analysis. From Fig. 2 (a) and (b), it is evident that the optimal control policies obtained through the use of DAEs, QP, and LS methods are identical. This observation validates the effectiveness of propositions 4.2 and 4.3 empirically.

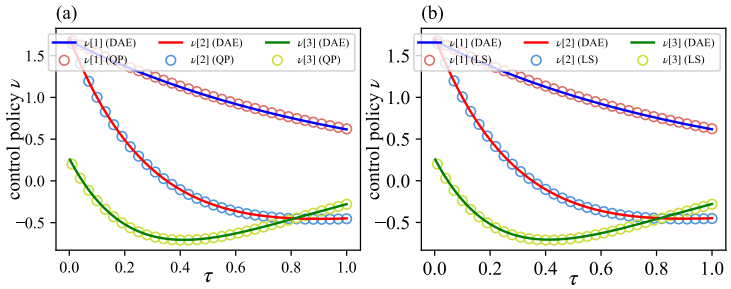

Figure 2 | The comparison of (a) OC policy of (43) solved by DAE and QP; (b) OC policy of (43) solved by DAE and LS.

5.3. Efficiency of OC policy Computation

The previous subsection empirically demonstrated the effectiveness of LS and QP problems. In this subsection, the focus is on investigating the efficiency of OC policy computation for the conversion. Building upon the problem defined in (43), the problem scales are expanded by altering the sampling noise, e. *Specifically, sampling* e *32 times results in 32 distinct problems, effectively increasing the data volume to 32.* The results are plotted in Fig. 3.

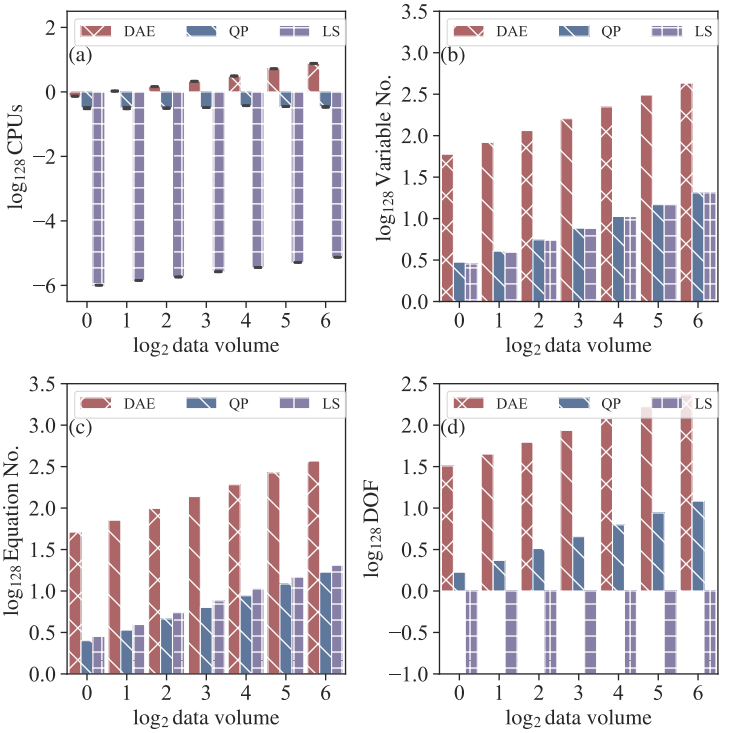

Figure 3 | The comparison of (a) computation time, (b) variable number, (c) equation number, and (d) DOF ($\log_{128}(0)$ is substitute as -1) along data volume.

Fig. 3(a) demonstrates that the computation times for the QP strategy are significantly lower than those for the DAE strategy, with the LS strategy showing even greater efficiency compared to the QP strategy. This observation confirms that our discretization-free OC computation strategies, based on QP and LS, are highly efficient and empirically substantiate our time complexity analysis. Figs. 3(b)

and (c) elucidate the reasons for this efficiency. As data volume increases, the number of variables and equations in the DAE strategy grows substantially, whereas the problem scale for QP and LS strategies remains relatively stable. Furthermore, Fig. 3(d) reveals that the degrees of freedom (DOF) for QP and LS follow a similar pattern, with LS having zero DOF. This suggests that the QP and LS conversions effectively reduce the exploration space. Collectively, Figs. 3(b)-(d) indicate that efficiency improvements primarily arise from reduced problem scale and DOF, thereby empirically supporting Remark 4.2 and Proposition 4.1.

Table 1 | Comparison Results on Different Datasets

Models	LDE-NPLVM		PPCA		PFA		Stein PCA		Batch Sinkhorn	
Metrics	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
abalone	**6.629E+00**	**1.835E+00**	8.295E+00	2.065E+00	1.003E+01	2.245E+00	1.005E+01	2.261E+00	7.286E+00	1.912E+00
bodyfat	**2.029E−05**	**3.362E−03**	9.695E−05	8.092E−03	4.284E−04	1.709E−02	4.258E−04	1.707E−02	6.610E−05	5.956E−03
mpg	**4.036E+01**	**5.145E+00**	7.509E+01	7.212E+00	1.368E+02	1.004E+01	1.336E+02	9.912E+00	5.790E+01	6.266E+00
simulation	**1.383E−01**	**2.880E−01**	1.056E+00	8.835E−01	1.984E+00	1.208E+00	1.993E+00	1.211E+00	3.641E−01	4.693E−01
space_ga	6.402E-02	2.025E-01	**2.845E−02**	**1.343E−01**	4.424E−02	1.741E−01	4.523E−02	1.764E−01	2.125E−02	1.100E−01

[1] **Bolded** results indicate the best value.

5.4. Model Performance on Regression Task

In this subsection, we want to answer the question about 'Does integrating a diffusion process into LLVM improve performance in downstream tasks?'. The algorithm for LDE-NLVM on regression task is given in Section SII.A Algorithm S1 of Supplementary Material. We assess the different models using evaluation metrics such as root-mean squared error (RMSE) and mean absolute error (MAE) defined as follows to evaluate the model performance:

$$\mathrm{RMSE} = \frac{1}{\mathrm{N_{test}}} \sum_{i=1}^{\mathrm{N_{test}}} \sqrt{(\hat{y}_i - y_i)^2}, \tag{44}$$

$$\mathrm{MAE} = \frac{1}{\mathrm{N_{test}}} \sum_{i=1}^{\mathrm{N_{test}}} |\hat{y}_i - y_i|, \tag{45}$$

We conduct experiments on five datasets namely abalone, bodyfat, mpg, simulation, and space ga. Simulation datasset is numerical dataset generated by ourselves, and the generated detailed can be referred to Section SII.A in supplementary material. Other datasets are public dataset dowloaded from LIBSVM and UCI websites . For fairness, we choose conventional latent varaible model named variational Bayesian probabilistic principal analysis (VB-PPCA) [11] and variational Bayesian probabilistic factor analysis (VB-PFA) [11] as baseline models. Besides, we also chosse recent optimal transport-based methods namely Stein Probabilistic Principal Analysis (Stein-PCA) [15] and Batch Sinkhorn Imputation [16] (state-of-the-art model) as our baseline models. Main results are reported in Table 1. Experimental Details can be found in Section SII.C in supplementary material.

From Table 1, it can be observed that our LDE-NLVM model achieves a reduction of 9.02~34.02% and 4.03~18.85% in terms of MSE and MAE metrics, respectively, compared to the baseline models. Overall, our LDE-NLVM model outperforms the baseline models for most of the datasets, indicating its superior performance. This phenomenon demonstrates that our model is more effective than conventional linear latent variable models for regression tasks thanks to the introduction of nonlinear function.

5.5. Model Convergence Analysis

In this subsection, we aim to support our claim regarding model convergence and answer question 'Does the adjoint EM algorithm converge?'. Based on the experimental result on model performance, we further adjust the learning rate on simulation dataset and plot the cost of the control policy $\int_0^1 \|\nu\|_2^2 d\tau$, and the log-likelihood term against training epochs in Fig. 4 (a) and (b), respectively. From Fig. 4, it can be observed that our model eventually converges. This observation further validates our proposition regarding algorithm convergence in practice. Additionally, in some cases with higher learning rates, our model is able to converge within 10 steps, indicating a potentially fast convergence rate for our algorithm. Overall, we provide evidence to support the theoretical and practical convergence properties of our parameter estimation approach for the LDE-NLVM model.

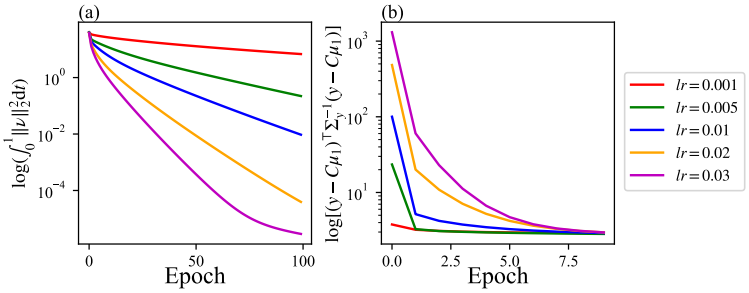

Figure 4 | The convergence line of (a) likelihood term; (c) control policy over **simulation dataset**.

6. Conclusions

In this paper, a novel model named LDE-NPLVM and its parameter learning algorithm, named Adjoint EM, were proposed. These were developed with consideration of the data generative process as nonlinear, albeit represented through simple LDEs. The learning objective was formulated through an in-depth analysis of the diffusion process in LVMs, and the learning algorithm was derived from optimization and optimal control principles. Notably, two equivalent programming problems were identified, enabling the efficient attainment of the optimal control law, in accordance with Pontryagin's maximum principle. Furthermore, gradients with respect to model parameters were derived, utilizing the adjoint state obtained in the E-step. The paper also presents the proposed parameter learning algorithm and its convergence analysis. A series of experiments were conducted to validate the effectiveness of the proposed model. Future research directions could include transforming our algorithm into a particle form and exploring the relaxation of the linear assumption, which presents an intriguing area of investigation.

References

[1] Radford M Neal and Geoffrey E Hinton. A view of the em algorithm that justifies incremental, sparse, and other variants. *Learning in graphical models*, pages 355–368, 1998.

[2] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society: series B (methodological)*, 39 (1):1–22, 1977.

[3] Stephen Boyd, Neal Parikh, Eric Chu, Borja Peleato, Jonathan Eckstein, et al. *Distributed*

optimization and statistical learning via the alternating direction method of multipliers, volume 3. Now Publishers, Inc., 2011.

[4] Francisco Vargas. Machine-learning approaches for the empirical schrödinger bridge problem. Technical report, University of Cambridge, Computer Laboratory, 2021.

[5] Qinsheng Zhang and Yongxin Chen. Path integral sampler: A stochastic control approach for sampling. In *International Conference on Learning Representations*, pages 1–13, 2021.

[6] Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. Ffjord: Free-form continuous dynamics for scalable reversible generative models. In *International Conference on Learning Representations*, pages 1–10, 2018.

[7] Simo Särkkä and Arno Solin. *Applied stochastic differential equations*, volume 10. Cambridge University Press, 2019.

[8] Bethany Nicholson, John D Siirola, Jean-Paul Watson, Victor M Zavala, and Lorenz T Biegler. pyomo. dae: A modeling and automatic discretization framework for optimization with differential and algebraic equations. *Mathematical Programming Computation*, 10:187–223, 2018.

[9] Xiaocan Li, Shuo Wang, and Yinghao Cai. Tutorial: Complexity analysis of singular value decomposition and its variants. *arXiv preprint arXiv:1906.12085*, 2019.

[10] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[11] Christopher M Bishop and Nasser M Nasrabadi. *Pattern recognition and machine learning*, volume 4. Springer, 2006.

[12] John Charles Butcher. *Numerical methods for ordinary differential equations*. John Wiley & Sons, 2016.

[13] Lorenz T Biegler and Victor M Zavala. Large-scale nonlinear programming using ipopt: An integrating framework for enterprise-wide dynamic optimization. *Computers & Chemical Engineering*, 33(3):575–582, 2009.

[14] William E Hart, Jean-Paul Watson, and David L Woodruff. Pyomo: modeling and solving mathematical programs in python. *Mathematical Programming Computation*, 3:219–260, 2011.

[15] Qiang Liu and Dilin Wang. Stein variational gradient descent: A general purpose bayesian inference algorithm. *Advances in neural information processing systems*, 29, 2016.

[16] Boris Muzellec, Julie Josse, Claire Boyer, and Marco Cuturi. Missing data imputation using optimal transport. In *International Conference on Machine Learning*, pages 7130–7140. PMLR, 2020.